Google data analytics capstone

bicycle-share company analysis

By Adi Grinshpan

Contents:

Introduction:

Cyclistic is a fictional bicycle-share company located in Chicago, IL.

The director of marketing believes the company's future success depends on maximizing the number of annual memberships.

Goal: to give my recommendations, based on the following analysis, on how to convert casual riders into annual members.

Ask phase:

The questions that will guide this analysis:

- How do annual members and casual riders use Cyclistic bikes differently?
- Why would casual riders buy Cyclistic annual membership?
- How can Cyclistic use digital media to influence casual riders to become members?

Tools: I'll be using **MySQL Shell** and **MySQL Workbench** for analysis, And **Tableau** for visualization.

Prepare phase:

We know that a good data is ROCCC: Reliable, Original, Comprehensive, Current and Cited.

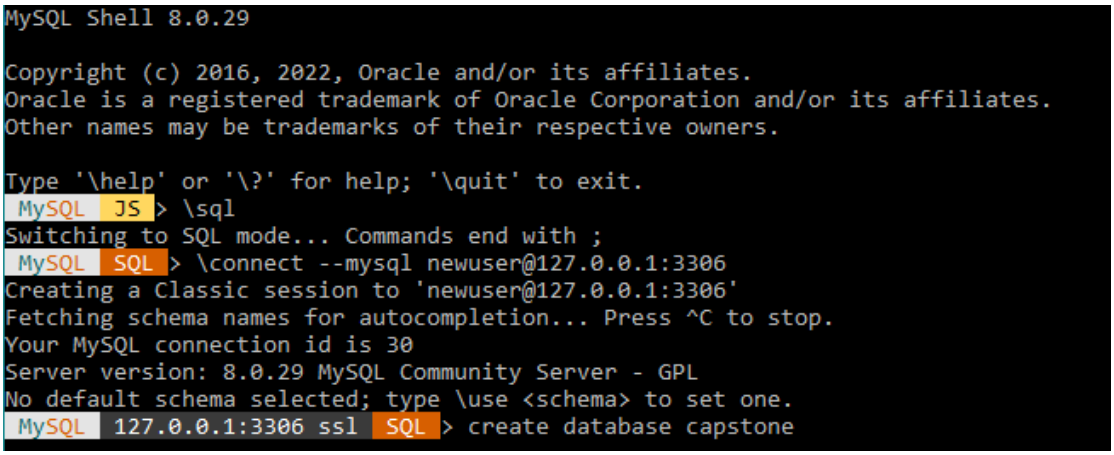
The data for this analysis is supplied by a real company named 'Motivate International Inc.', it is uploaded and stored on a monthly basis to Index of bucket "divvy-tripdata" .

The latest 24 months of data, spanning **between April 2020 and March 2022**, are available and used for this analysis.

Each file consists of the following columns:

- Ride id: a unique key for each ride
- Rideable type: the kind of bike used for the ride (docked bike, electric bike, classic bike)
- Started at: a date/time for the beginning of the ride
- Ended at: a date/time for the ending of the ride
- Start station name
- Start station id
- End station name
- End station id
- Start lat: latitude of the beginning of the ride
- Start lng: longitude of the beginning of the ride
- End lat: latitude of the ending of the ride
- End lng: longitude of the ending of the ride
- Member/casual: the type of user (member or casual)

Creating database for the analysis using MySQL Shell:



*Switching to **MySQL Workbench**

Creating the table with the columns I'll be using for the analysis:

```
create table cyclistic (
ride_id varchar (65),
rideable_type ENUM('classic_bike','electric_bike','docked_bike'),
started_at DATETIME,
ended_at DATETIME,
start_station_id varchar (65),
end_station_id varchar (65),
member_casual ENUM('member','casual')
) ;
```

uploading the data files one-by-one into the table:

```
LOAD DATA LOCAL INFILE  *file path*  INTO TABLE cyclistic
FIELDS TERMINATED BY ',' LINES TERMINATED BY '\r\n'  IGNORE 1 LINES
@)col1,@col2,@col3,@col4,@col5,@col6,@col7,@col8,@col9,@col10,@col11,@col12,@col13,
@col14 (
set
ride_id=@col1,rideable_type=@col2,started_at=@col3,ended_at=@col4,start_station_id=
@col6,end_station_id=@col8,member_casual=@col13 ;
```

creating a duplicate table for backup:

```
create table cyclistic2 (
ride_id varchar (65),
rideable_type ENUM('classic_bike','electric_bike','docked_bike'),
started_at DATETIME ,
ended_at DATETIME ,
start_station_id varchar (65),
end_station_id varchar (65),
member_casual ENUM('member','casual')
) ;
```

populating the duplicate table with the data from the original one:

```
insert into cyclistic2(ride_id, rideable_type, started_at, ended_at, start_station_id,
end_station_id, member_casual)
select ride_id, rideable_type, started_at, ended_at, start_station_id, end_station_id,
member_casual from cyclistic;
```

Process phase:

A glimpse at the data:

```
select * from cyclistic limit 5;
```

output:

ride_id	rideable_type	started_at	ended_at	start_station_id	end_station_id	member_casual
A847FADBBC638E45	docked_bike	2020-04-26 17:45:14	2020-04-26 18:12:03	86	152	member
5405B80E996FF60D	docked_bike	2020-04-17 17:08:54	2020-04-17 17:17:03	503	499	member
5DD24A79A4E006F4	docked_bike	2020-04-01 17:54:13	2020-04-01 18:08:36	142	255	member
2A59BBDF5CDBA725	docked_bike	2020-04-07 12:50:19	2020-04-07 13:02:31	216	657	member
27AD306C119C6158	docked_bike	2020-04-18 10:22:59	2020-04-18 11:15:54	125	323	casual

The period for study:

```
select max(started_at) as period_start,
```

```
min(started_at) as period_end,
datediff (max(started_at), min(started_at)) as period_in_days
from cyclistic;
```

output:

period_start	period_end	period_in_days
2022-03-31 23:59:47	2020-04-01 00:00:30	729

Checking for duplications:

```
select count(ride_id) as 'count_ID',
count(distinct(ride_id)) as 'count_distinct_ID' ,
(Count(ride_id))-(count(distinct(ride_id))) as 'duplicates' from cyclistic;
```

Output:

	count_ID	count_distinct_ID	duplicates
▶	9213280	9213071	209

Although the number of duplications is negligible, I decided -for practice purposes - to do a quick check by selecting the duplications for further inspection:

```
with cte_dup AS (
select * , count(*) over (partition by ride_id) as 'count' from cyclistic2)
select * from cte_dup where count>1 ;
```

Output:

	ride_id	rideable_type	started_at	ended_at	start_station_id	end_station_id	member_casual	count
▶	021A73F8C18B932D	docked_bike	2020-12-15 12:15:58	2020-11-25 16:48:02	TA1309000035	TA1309000018	member	2
	021A73F8C18B932D	docked_bike	2020-11-25 16:35:39	2020-11-25 16:48:02	325	314	member	2
	0334987B57662109	docked_bike	2020-11-25 16:15:04	2020-11-25 16:22:04	294	459	member	2
	0334987B57662109	docked_bike	2020-12-15 11:56:33	2020-11-25 16:22:04	13109	KA1504000152	member	2

We can see that the duplications have the 'started at' after the 'ended at'.

with the knowledge of having a backup table, I can confidently delete the duplications:

```
delete from cyclistic  where ride_id IN (
        select ride_id from (select ride_id , row_number ()
                over(partition by ride_id order by ride_id) as 'duplicates'
        from cyclistic) as temp_t where duplicates>1 AND started_at > ended_at ) ;
```

Checking for NULLs and other forms of missing values:

```
select * from cyclistic where
ride_id is null
OR rideable_type is null
OR started_at is null
OR ended_at is null
OR start_station_id is null
OR end_station_id is null
OR member_casual is null;
```

No NULL observations were found.

Checking for missing values that are “”:

```
select * from cyclistic where ride_id=""  OR rideable_type= ""  OR member_casual="" ;
```

no missing values found in these columns either.

I did find missing values in 2 columns – “start_station_id” and “end_station_id”

```
select count (*) from cyclistic where start_station_id="";
select count (*) from cyclistic where end_station_id="";
```

start_station_id had 868,174 missing values and end_station_id had 939,949 missing values.

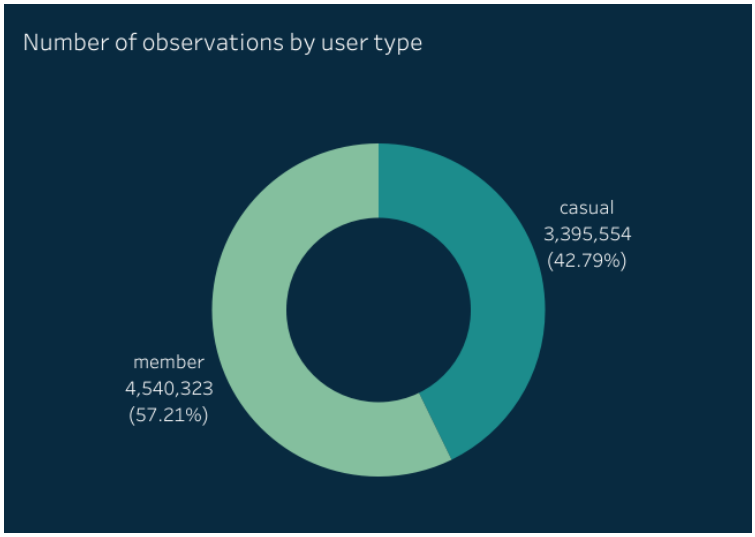
*I decided to drop these observations based on further analysis added in the appendix.

```
delete from cyclistic where start_station_id="" OR end_station_id="" ;
```

Analyze and Share phases:

number of observations by user type:

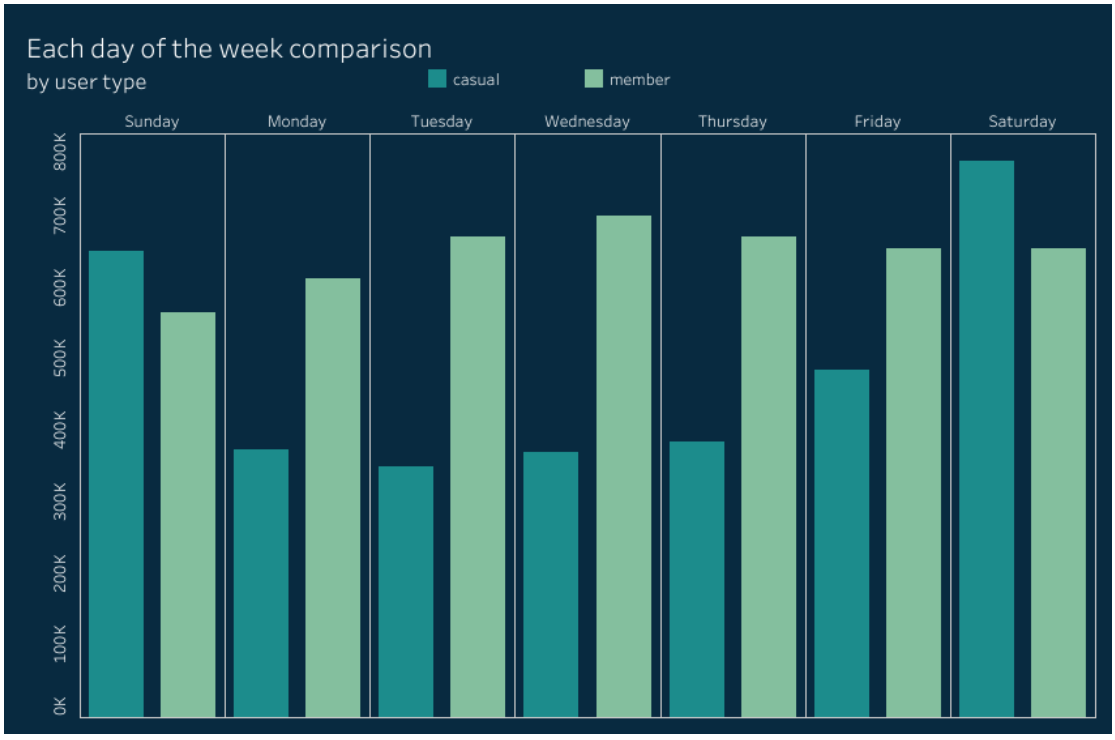
```
select member_casual , count(member_casual) as "sum"   from cyclistic group by 1 ;
```



As can be seen, during the period in scope (24 months), the majority of the rides (57%) were done by members. It's the rest (43%) we're looking to convert to membership users.

Looking for differences between the user type during the week:

```
SELECT member_casual
            , dayofweek(started_at) AS most_active_day_of_week
  , count(1) AS total
 FROM cyclistic
GROUP
  BY dayofweek(started_at)
  , member_casual
  order by 1,3 desc ;
```

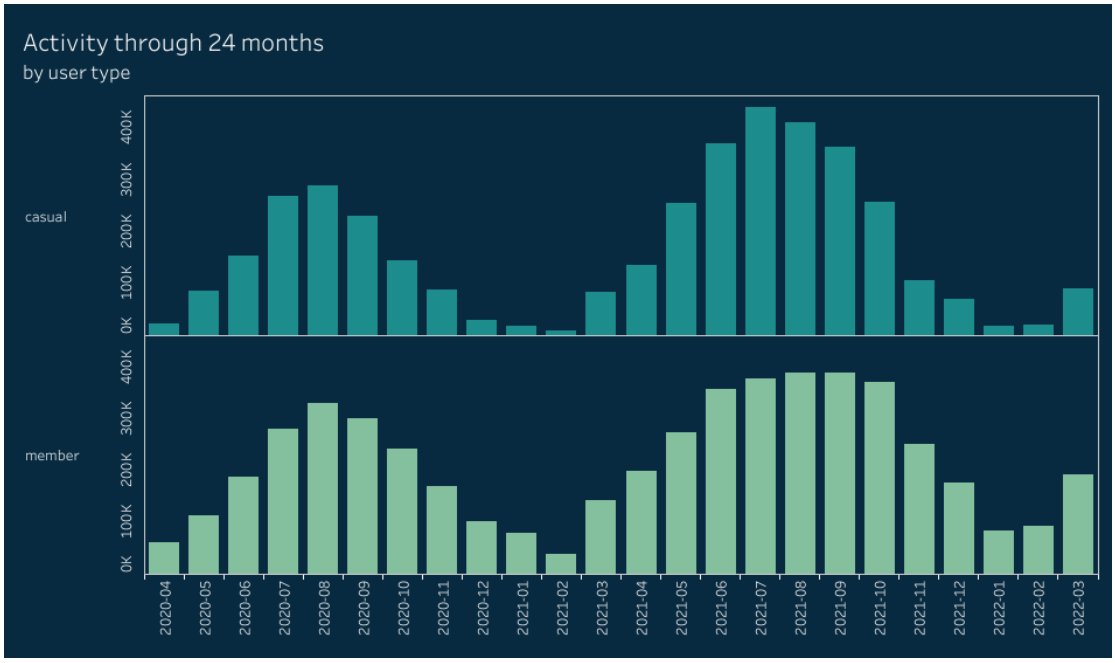


there are differences in activity days.

For the members, the activity is relatively stable throughout the week with Wednesday the most active day.

For the casual users, we can see a spike in activity during the weekends -Saturday and Sunday.

Looking for seasonality differences between user types:



We can see that throughout the 2-year period, the activity of members and casual users don't differ significantly from one another. For both we can see a spike in activity during July to September period and lower activity during the winter months (December to March).

Looking for median ride duration differences between user types:

In order to avoid outliers, I decided to find the median ride duration for each user type (instead of average) using the following method -

```
select count(member_casual) from cyclistic where member_casual = 'member' ;
```

output: 4540323

```
select count(member_casual) from cyclistic where member_casual = 'casual'  ;
```

output: 3395554

```
with cte AS (

select member_casual ,TIMEDIFF(ended_at ,started_at) as "duration" ,row_number()
over(partition by member_casual order by "duration" ) as "ranks"  from cyclistic)


select member_casual ,duration,ranks from cte where member_casual = "member" and
ranks = ceil(4540323/2)

UNION

select member_casual ,duration,ranks from cte where member_casual = "casual" and ranks
= ceil(3395554/2);
```

output:

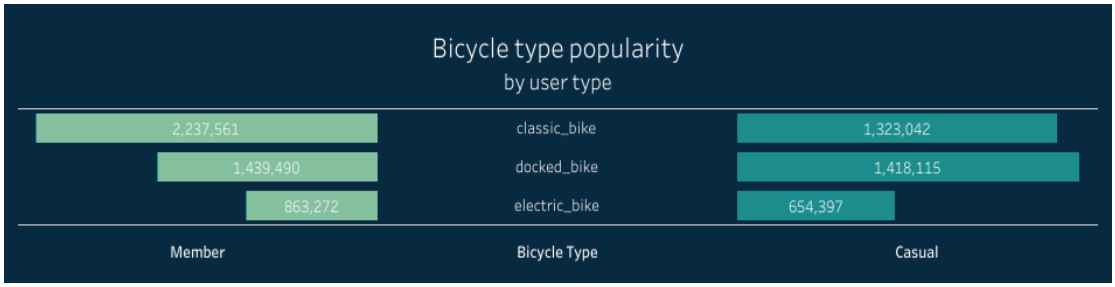
member_casual	duration	ranks
member	00:09:16	2270162
casual	00:27:15	1697777

As can be seen, the median ride duration for a member is 9 minutes, which is 3 times shorter than the median ride of a casual user (27 minutes). This finding might add to the former finding that showed casual users are more active during the weekend -so it makes sense their ride duration is longer -i.e., for leisure usage and not for work usage.

Favorable bicycle type for each user type:

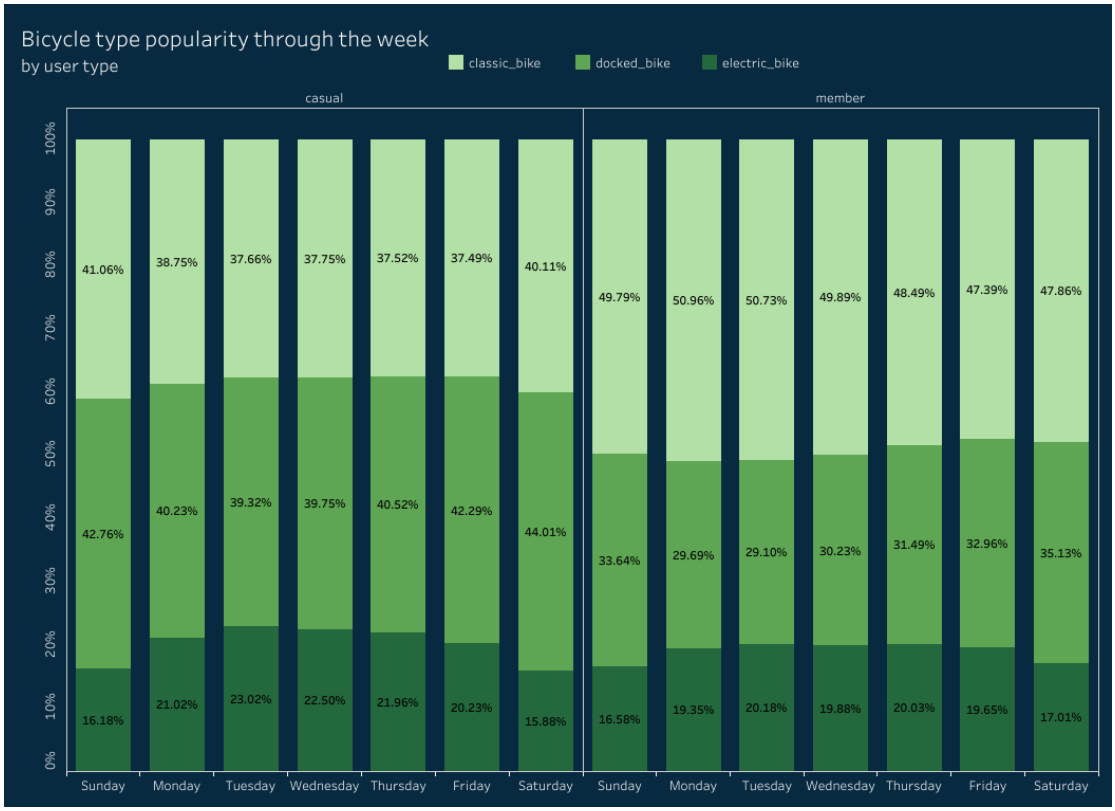
```
select member_casual ,

rideable_type ,

count(rideable_type) as "count",
```

```
row_number()over(partition by member_casual order by count(rideable_type) desc) as "ranking"

from cyclistic group by 1,2 ;
```



For the members, the classic bike is the most popular type. For the casual users the docked bike the most popular. For both, the electric bike is the least popular type.

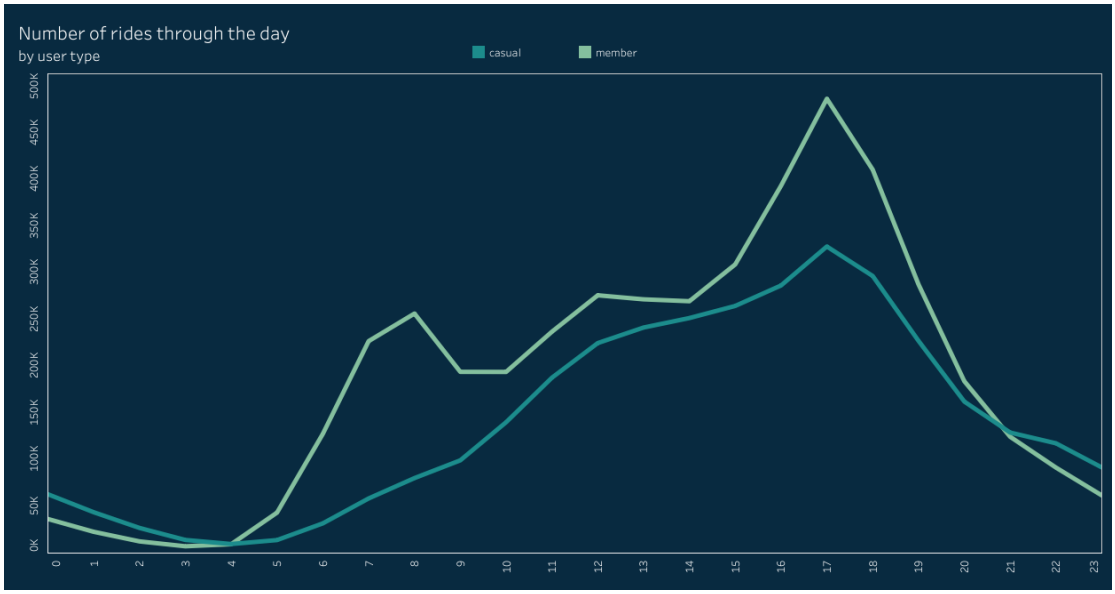
This can be seen as well when we zoom-in on week-day distribution -



Looking for differences in activity through the day between user types:

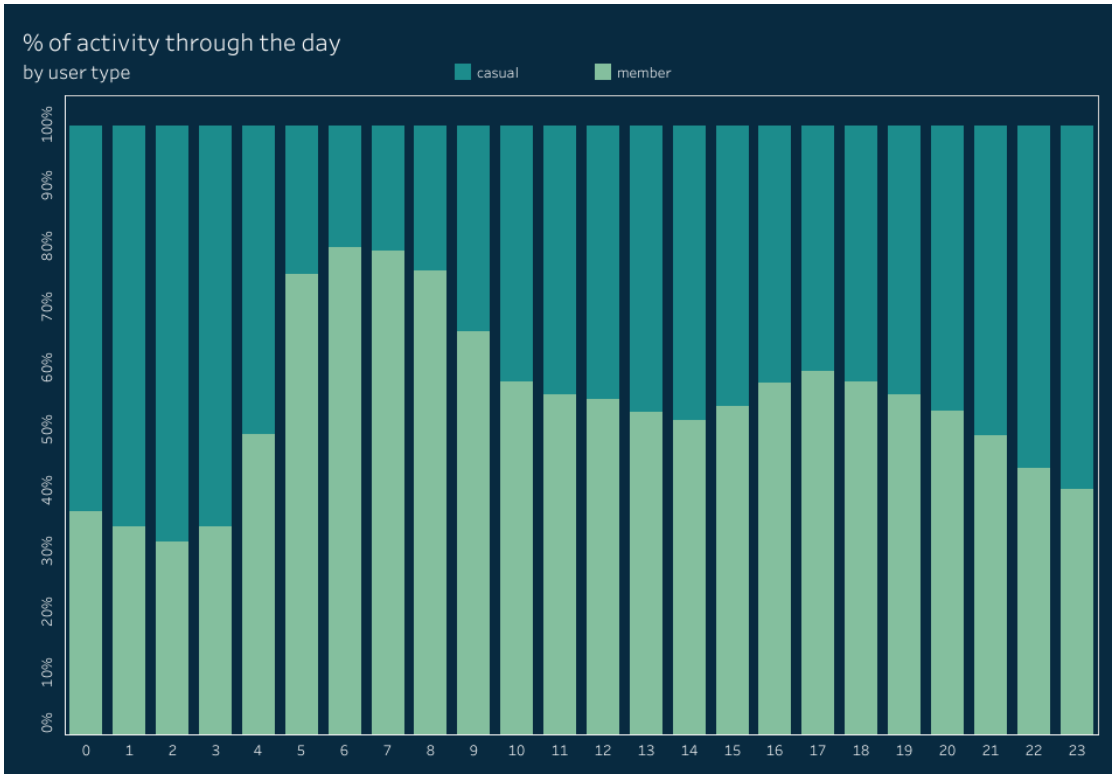
```
select member_casual ,

hour(started_at) ,

count(extract(hour from started_at))
```

```
from cyclistic group by 1,2 order by 1,3 desc;
```



We can see that the members have 2 peaks in activity -one in the morning around 8 AM and the other in the afternoon around 5 PM.

The casual users have one peak in the afternoon, around 5 PM.

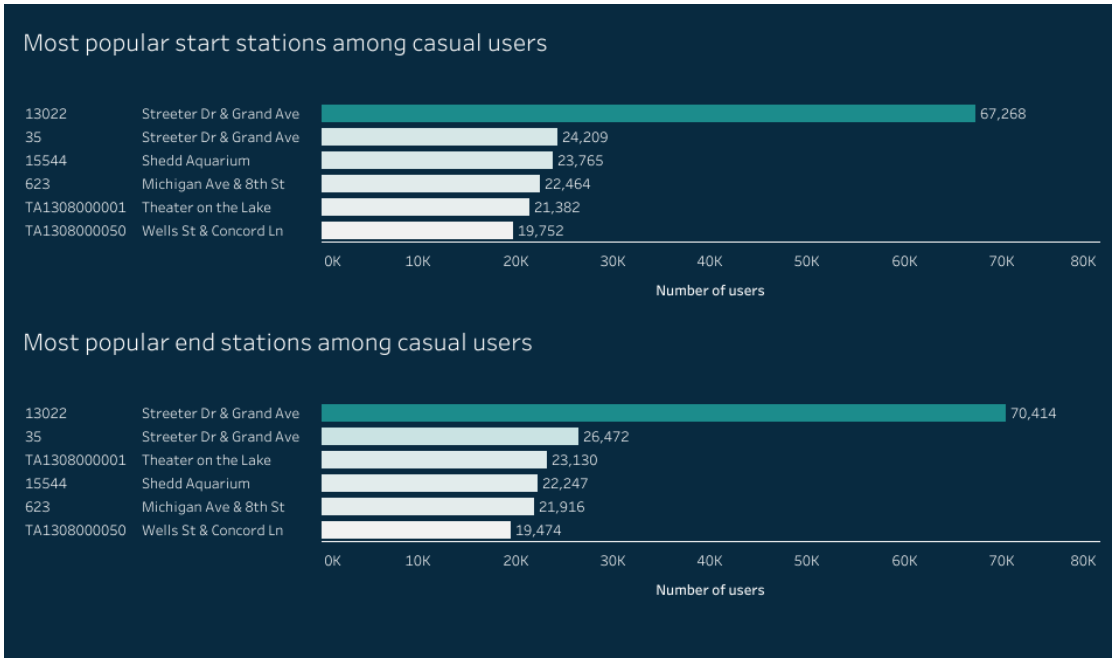


Here we can see the distribution between the user types through the day in percent.

During the morning-commute hours (5 AM to 9 AM), the vast majority of users are members, as the day progress we see a more equal distribution between user types.

Favorable stations for casual users

I would like to see the most popular start and end stations for casual users for targeting physical advertisement.



We can see the most popular stations for picking the bike -are the same for returning the bike at the end of the journey, yet some in different order.

To summarize the analysis and share phases:

During the 24 months in scope, 43% of the rides were done by casual users.
The casual users don't differ significantly from members on a month-to-month overview - meaning, for both user types we can see a peak in activity during July to September, and lower activity during the winter months (December to March).

During the week, casual users' activity is mostly during the weekends -Saturday and Sunday, as oppose to members with stable activity through the week.

Casual users also prefer Docked bikes and Classic bikes -that preference does not change much during the week. The casual users have one peak in activity during the day in the afternoon, at around 5 PM.

We've also learned, which stations see more casual users' activity.

Act phase (recommendations):

- Since the rise in activity starts for each user type in May rising to peak in August-September, we have this in-between period best for advertisement to increase awareness to the brand and service.

- Casual members are more active during the weekend (Saturday - Sunday), so we can create a membership campaign which encourages the customers to use bikes during working days by portraying the advantages, such as avoiding traffic and avoiding spending time looking for parking during the morning commute. Also, the weekend would be preferred time to push the adds.

- for physical advertisement, we can publish at the locations (start and end stations) most visited by the casual users.

Appendix:

Dropping observations where start_station_id OR end_station_id is “”

I decided on dropping these observations after reviewing their distribution.
In the following outputs, we can see that the distribution of the observations with the missing values is matching that of the total population in regards to each user type activity during the week, thus I concluded that dropping them will not affect the general analysis.

start station id = ""

member_casual	most_active_day_of_week	total
member	4	73516
member	5	72789
member	6	72740
member	3	72662
member	7	69457
member	2	66860
member	1	61414
casual	7	69862
casual	1	60361
casual	6	57600
casual	5	49334
casual	4	48459
casual	3	46952
casual	2	46168

end station id =""

member_casual	most_active_day_of_week	total
member	4	74976
member	5	74832
member	6	74601
member	3	74039
member	7	70375
member	2	67502
member	1	61821
casual	7	81258
casual	1	69826
casual	6	67388
casual	5	58016
casual	4	56910
casual	3	54800
casual	2	53604

total population

member_casual	most_active_day_of_week	total
member	4	805865
member	5	776655
member	3	775547
member	6	760784
member	7	755614
member	2	707531
member	1	653998
casual	7	885916
casual	1	745701
casual	6	573413
casual	5	460304
casual	4	445068
casual	2	444456
casual	3	422010